

EXHIBIT 72

LINFO

Source Code Definition

Source code (also referred to as *source* or *code*) is the version of *software* as it is originally *written* (i.e., typed into a computer) by a human in *plain text* (i.e., human readable alphanumeric *characters*).

The term *software* refers to all *operating systems*, *application programs* and data that is used by products containing microprocessors (also called *processors* or *central processing units*). Such products include not only personal computers but also a vast array of other products, such as aircraft electronic systems, railway signaling systems, industrial robots, electronic medical equipment, space vehicle guidance systems, electronic cameras and even simple electronic toys.

Source code can be written in any of the hundreds of programming languages that have been developed. Some of the most popular of these are C, C++, Cobol, Fortran, Java, Perl, PHP, Python and Tcl/Tk.

There are many *programs* that can be used for writing source code in the desired programming language, ranging from simple, general purpose text editors (such as *vi* or *gedit* on *Linux* or *Notepad* on Microsoft Windows) to *integrated development environments* (such as *Visual C++* on Microsoft Windows or the *cross-platform Eclipse Platform* for constructing and running integrated software-development tools). After writing, the source code is saved in a single file or, more commonly, in multiple files, with the number of files depending on such factors as the programming language and the size of the project.

Very often it is the case that source code intended for use on one platform can be written on another platform. A platform is any combination of processor and/or operating system. However, there are some notable exceptions. For example, the popular Microsoft text editors *Notepad* and *Wordpad* cannot be used to write code for deployment on Unix-like operating systems because they do not treat new lines of text correctly.

To be usable by a computer or other microprocessor-based product, source code must be *compiled* (i.e., translated by a computer) into *machine language* by a special program called a *compiler*. Also referred to as *object code* or *binaries*, machine language consists of a sequence of instructions (in the form of zeros and ones) that the processor can understand -- but which is very difficult for humans to read or modify.

A program's source code is not necessarily all written in the same programming language. For example, it is common for programs that are written primarily in C to contain portions that are in an *assembly language* for optimization of processor efficiency. This is the case with the Linux *kernel* (i.e., the core of the Linux operating system).

An assembly language is a human-readable notation for machine language for a specific processor. It is far easier for programmers to remember and write assembly language commands than it is the zeros and ones that constitute machine language because they are short, mnemonic series of alphanumeric characters. In contrast to *high-level*

programming languages such as C, Java and Perl, there is a close correspondence between assembly languages and machine languages.

Even moderately complex software can require the compilation of dozens, or hundreds, of separate source code files. However, this complexity can be reduced considerably by the inclusion of a *makefile*, which is a file that describes the relationships among the source code files and contains information about how they are to be compiled.

Uses For Source Code

Software is distributed (i.e., sold or given away) primarily in compiled form (i.e., binaries) rather than in source code form. There are good reasons for this, including the fact that it is usually much easier to install (particularly for ordinary users), it keeps the source code secret and the file sizes are much smaller (thereby speeding downloads from the Internet and conserving disk space).

As an example of the huge size differences in file size that can exist between the source and binary versions of a program, AbiWord (a popular, stand alone and *free* word processing program that is available for the Linux, Microsoft Windows and QNX operating systems) has a source code size of 25.3MB but a binary (RPM format) size of only 3.54MB (both for version 2.0.6 for Linux).

The contrast is even greater for the Linux kernel. For example, the source code for version 2.4 is approximately 100MB and contains nearly 3.38 million lines of code, whereas its compiled binary is only about 1.1MB on a typical system. (The size of the Linux kernel source code has been increasing nearly exponentially ever since its inception, soaring from 0.2MB and 10,239 lines of code for version 0.01 in 1991 to 212MB and 5.93 million lines of code for version 2.6.0 in 2003.)

Although its primary purpose is usually to generate ready-to-install binaries, source code has several other important uses as well, and thus there is often a demand for it along with, or even in place of, the precompiled versions of the software. The reasons for this relate to the fact that the source code is generally the most versatile, informative and permanent form of any software:

- (1) System administrators and other experienced users frequently prefer to compile software themselves prior to installation because of the increased number of options and consequent greater control that it can provide in tailoring the software to particular requirements.
- (2) It is relatively easy to fix *bugs* (i.e., errors), find viruses or other malicious content as well as to enhance or otherwise modify software using the source code, whereas it would be extremely difficult using the binaries.
- (3) Having the source code version of software makes it practical to *port* the software to other platforms (i.e., develop versions for other processors and/or operating systems). Without the source for a particular piece of software, such porting is usually extremely difficult.
- (4) Source code can be a valuable tool for learning, particularly for beginning programmers, who often find it helpful to study existing code to gain insight into programming techniques and methodology.
- (5) Source code also serves as a communication tool between experienced programmers. This is because of its (at least ideally) concise and unambiguous nature. The sharing of

source code between developers is frequently cited as a contributing factor to the refinement of their programming skills.

(6) Programmers often want source code because of their tendency to recycle parts of it into new projects, a practice referred to as *software reusability*.

When the source code for a software package has been lost or is otherwise unobtainable, it is possible to *reverse engineer* it in order to obtain the code. However, reverse engineering is not easy, and there are frequently legal and contractual restrictions. In particular, it constitutes a violation of copyright law if done for the sole purpose of copying or duplicating programs. Reverse engineering for any purpose is often prohibited by [EULAs](#) (end user license agreements).

Open Source Versus Closed Source

One of the most intensely debated topics in the computer field continues to be the relative merits of *open source* and *closed source* software. The former is software for which the source code is freely available (i.e., at no cost and easily accessible) for anyone to use for any purpose, including studying, modifying, extending, giving away or even selling. Although there are some philosophical differences behind the movements, in most cases open source software is basically the same as [free software](#).

The latter provide their source code either with the compiled software or by allowing it to be downloaded from the [Internet](#). The [GNU General Public License](#) (GPL), the most widely used free software license, actually makes it a legal requirement that the source code for all software released under it be made freely available to all users.

Closed source software is software for which the source code is kept secret. Most [proprietary](#) (i.e., commercial) software is closed source. There are several reasons that developers of proprietary software take great pains to keep their source code secret, including the concerns that:

- (1) Other developers might copy some of their code and use it in other programs.
- (2) Hackers will find vulnerabilities in the code that will enable them to develop viruses, spyware or other [malware](#) (i.e., malicious software) for it.
- (3) Public disclosure of the source code could expose its developers to charges that some of the code was plagiarized from other programs.
- (4) Customers will try to modify the source code, resulting in new problems that could be difficult for either the customers or developer to correct.
- (5) The source code could be used as evidence in legal proceedings, particularly those related to whether the developer has been complying with certain legislation or court decisions.
- (6) The source code could contain unfavorable *comments* inserted by programmers about their employer, customers or competitors. Comments are words, phrases, sentences or paragraphs that are interspersed in source code but which do not affect the operation of the code. Their official purpose is to [document](#) the code and explain it to other programmers (who may have to repair or revise the code at some future date), although they are frequently used for other purposes as well. It can be difficult to find and remove potentially offensive comments because of the great length of the code for large programs and the subtlety with which some comments are written.

Reasons to Not Keep Source Code Secret

There are also reasons for not keeping source code secret and for allowing, or even encouraging, others to view and study it. Advocates of open source point out that this approach makes it possible for a much larger and more diverse set of qualified people to examine the source code, thus resulting in the discovery of more bugs and providing more and better suggestions for improvements and extensions.

That this approach has been extremely successful is evidenced by the rapid improvements in performance of numerous open source programs, many of which are equal to or superior to their closed source counterparts. One of the most outstanding examples is the Apache web server, which is currently hosts more than 70 percent of all web sites on the Internet.

Undoubtedly the most famous example is Linux, the use of which is continuing to grow rapidly for a wide range of applications, including supercomputers, enterprise computing systems, personal computers and embedded systems. Another example is the [GNU Compiler Collection](#) (GCC), which contains very highly rated and widely used compilers for C, C++, Fortran, Java and other programming languages. In fact, the concept of open source is so appealing that there are currently thousands of open source projects in various stages of development.

Some advocates of open source contend that source code be considered a legally protected form of free speech, i.e., that it should not be illegal to publish source code obtained by reverse engineering or other legal means. An example of where this is particularly controversial is the source code for the encryption [algorithms](#) for DVDs and electronic books. Several cases have reached the courts, but as yet no definitive precedents have been set.

"Shared Source"

There is an alternative to closed source and open source. It is a compromise that attempts to attain the advantages of both approaches. Microsoft is using this alternative for some of its software and has dubbed it *shared source*.

Under the company's *Shared Source Initiative*, a few outsiders, chiefly security researchers, universities and government agencies, are allowed to view selected portions of the code under tightly restricted conditions and after signing strict nondisclosure agreements.

Shared source is seen by some as being a strategic move to address concerns by corporations and governments (both domestic and foreign) that there are security holes and [backdoors](#) in the software. A backdoor is intentionally inserted secret code that allows its developer or a government agency to search or modify software on a remote computer without the knowledge or permission of the owner of that computer. The unavailability of the source code for Microsoft software has been a major source of concern to foreign (i.e., non-U.S.) governments, which have feared that it could compromise their security. The availability of the source code for Linux, FreeBSD, Apache and other free software is thus viewed as another major advantage of such software.

Occasionally, portions of the source code for some proprietary software are *leaked* to the Internet. This could be evidence of the frequently made assertion by proponents of open source that it is difficult to keep source code secret, and thus the use of code secrecy as a means of providing security is unreliable.

However, it is possible that some such leaks have, in fact, been made intentionally by the developers themselves. One motive for this is that it would allow them to receive comments about bugs and other problems with selected sections of the code from a far larger and more diverse group of people than would be possible in-house while still preventing others from legally copying or reusing the code.

Created May 23, 2004. Last updated February 14, 2006.
Copyright © 2004 - 2006 The Linux Information Project. All Rights Reserved.